

The Complexity of Computing a Fourier Perturbation

Nir Ailon* Gal Yehuda †

Department of Computer Science
Technion Israel Institute of Technology
Haifa, Israel

April 12, 2016

Abstract

The complexity of computing the Fourier transform is a longstanding open problem. Very recently, Ailon (2013, 2014, 2015) showed in a collection of papers that, roughly speaking, a speedup of the Fourier transform computation implies numerical ill-condition. The papers also quantify this tradeoff. The main method for proving these results is via a potential function called quasi-entropy, reminiscent of Shannon entropy. The quasi-entropy method opens new doors to understanding the computational complexity of the important Fourier transformation. However, it suffers from various drawbacks. This paper, motivated by one such drawback, eliminates it by extending the method.

The argument goes as follows: If instead of computing the Fourier transform Fx of input $x \in \mathbb{R}^n$ we were to compute a Fourier ε -perturbation, defined as $(\text{Id} + \varepsilon F)x$, then the quasi-entropy method in the well-conditioned regime would, without any adjustments, lead to a linear algebraic lower bound of $\Omega(\varepsilon^2 n \log n)$ many operations (counting additions and multiplications on scalar variables). Had this bound been matched by an algorithm, then we would have been able to extract Fx in time $O(\varepsilon^2 n \log n)$ by first computing $(\text{Id} + \varepsilon F)x$, then subtracting x from the output and dividing the result by ε . By taking $\varepsilon = \Theta(1/\sqrt{\log n})$ we could artificially drive the computation time toward the trivial linear time lower bound. Such a scheme would suffer on a real computer from numerical errors, but this could be avoided by extending the computer word size by only $\Theta(\log \varepsilon^{-1}) = \Theta(\log \log n)$ bits. The end result is a Fourier algorithm in running time $\tilde{O}(n \log \log n)$ (counting logical bit operations, and using fast integer multiplication).

We generalize the quasi-entropy method so as to show that driving ε down does not allow such a free ride in case of the Walsh-Hadamard Fourier transform, and that the linear algebraic lower bound is, in fact, $\Omega((n \log n)/\log \varepsilon^{-1})$. This exactly ‘cancels out’ the numerical accuracy overhead. It also strengthens our belief that, roughly speaking, Fourier computation requires $\Omega(n \log n)$ time in a computational model that takes into account numerical accuracy and logical bit operations.

*nailon@cs.technion.ac.il

†gal2016@gmail.com

1 Introduction

The Fourier transform is one of the most important linear transformations in science and engineering. The (normalized) Discrete Fourier Transform (DFT) $\hat{x} \in \mathbb{C}^n$ for input signal $x \in \mathbb{C}^n$ is defined by

$$\hat{x}_i = n^{-1/2} \sum_{j=1}^n e^{-2\pi i \iota(i-1)(j-1)/n} x_j ,$$

where $\iota = \sqrt{-1}$. DFT has applications in many fields, including fast polynomial multiplication [12, chapter 30], fast integer multiplication [18], fast large scale linear algebra and matrix sketching [4, 27], signal processing [15, chapters 6-9], [22, 29] and more. From a theoretical perspective, the DFT is a special case of the more general Fourier transform on abelian groups, with respect to the group $\mathbb{Z}/n\mathbb{Z}$. Another special case, known as the Walsh-Hadamard transform, is defined over the group $(\mathbb{Z}/2\mathbb{Z})^{\log_2 n}$ (for integer $\log_2 n$). The Walsh-Hadamard transform \hat{x}^{WH} is given by

$$\hat{x}_i^{WH} = n^{-1/2} \sum_{j=1}^n (-1)^{\langle i-1, j-1 \rangle} x_j ,$$

where $\langle a, b \rangle$ here is dot product of two bit vectors corresponding to the base-2 representation of integers a, b . The WH transform has applications in coding theory and digital image processing [5, 7, 14, 21, 28] as well as in fast large scale linear algebra and matrix sketching [10, 24]. It is also instrumental in analysis of boolean functions [8, 9, 17] and more generally in theoretical computer science and learning theory [13, 23].

From a computational point of view, an $O(n \log n)$ algorithm is known for both the DFT and the WH transform. For the DFT case, this was discovered by Cooley and Tukey in 1965 [11] in a seminal paper. For the WH case, the corresponding WH transform has been discovered in 1976 [16]. Both algorithms run in a linear algebraic computational model (more on that in Section 2).

The complexity of computing the Fourier transform is, on the other hand a longstanding open problem. An $\Omega(n)$ bound is trivial due to the necessity to consider all input coordinates. The gap between $\Omega(n)$ and $O(n \log n)$ may seem small. Nevertheless, owing to the importance of the Fourier transform (both DFT and WH), it is crucial to close it in a reasonable model of computation. Some early results [25] proved a lower bound of the *unnormalized* Fourier transform, defined as a scaling up of the Fourier transform by $n^{1/2}$, using a potential function that is related to matrix determinant. The result used the fact that scaling up an n -by- n matrix by a factor of α increases its potential by a factor of α^n , while any linear algebraic operation involving numbers of at most constant modulus (as used in FFT) can increase the potential by a constant factor. This result, though shedding light on an important problem, unfortunately does not explain why the normalized (unitary) Fourier transform has complexity $\Omega(n \log n)$ and, conversely, does not explain why the Fourier transform is computationally more complex than a scaling of the identity matrix by a factor $n^{1/2}$. A result by Papadimitriou [26] provides a lower bound of computing the Fourier transform in finite fields, in a computational model that is not comparable with ours.

Recently Ailon [1, 2, 3] showed in a collection of papers that speedup of Fourier computation (both DFT and WH) implies ill-conditioned computation (see also Section 2 below for a precise definition). The result uses a potential reminiscent of Shannon entropy function on probability vectors, except that it is applied to any real vector (including e.g. negative numbers).

1.1 The Quasi-Entropy Method

In this work we work over the reals \mathbb{R} , and leave the DFT (which is complex) to future work. We briefly remind the reader of matrix quasi-entropy: For a nonsingular real n -by- n matrix M , the matrix quasi-entropy $\Phi(M)$ is given as

$$\Phi(M) = - \sum_{i=1}^n \sum_{j=1}^n M(i, j) M^{-T}(i, j) \log |M(i, j) M^{-T}(i, j)| . \quad (1.1)$$

where M^{-T} is shorthand inverse-transpose. The main result in [3] is, that as long as the condition number of M is at most κ , a *planar rotation* operation applied to a pair of rows of M can change entropy by at most $O(\kappa)$. A planar rotation matrix, or simply a rotation matrix for brevity, is obtained by multiplying M on the left with the (i, i') -plane Θ -rotation matrix $R_{i, i', \Theta} \in \mathbb{R}^{n \times n}$ defined by the identity $\begin{pmatrix} R_{i, i', \Theta}(i, i) & R_{i, i', \Theta}(i, i') \\ R_{i, i', \Theta}(i', i) & R_{i, i', \Theta}(i', i') \end{pmatrix} = \begin{pmatrix} \cos \Theta & \sin \Theta \\ -\sin \Theta & \cos \Theta \end{pmatrix}$, the remaining diagonal elements 1, and then the remaining elements 0.

The quasi-entropy of the n -by- n identity matrix Id_n is 0, that of the Fourier matrix (both DFT and WH) is $\Omega(n \log n)$. Therefore, in the uniformly κ -well conditioned model (see Section 2 for an exact definition) a lower bound of the number of steps is $\Omega(\kappa^{-1} n \log n)$. Equivalently, a speedup of FFT by factor of $\kappa > 1$ implies κ -ill conditioned computation. In fact, as argued in followup work [2], the κ -ill condition implication is very strong in the sense that at $\Omega(n)$ pairwise orthogonal directions in input space, at some point along the computation, $\Omega(\log \kappa)$ bits of information about each direction are lost due to overflow or underflow. We refer the reader to [2] for the precise statement, however, we need to use here an extension of the quasi-entropy function that was defined there. For two matrices A, B with n rows and n' columns each, the (A, B) -preconditioned quasi entropy of M , denoted $\Phi_{A, B}(M)$, is defined as

$$\Phi_{A, B}(M) = - \sum_{i=1}^n \sum_{j=1}^{n'} (MA)(i, j) (M^{-T}B)(i, j) \log |(MA)(i, j) (M^{-T}B)(i, j)| . \quad (1.2)$$

The reason we work with the preconditioned quasi-entropy function is that we will often track the change of the entropy as M evolves in the computation, while A, B remain fixed.

1.2 Problems with the Quasi-Entropy Method

(P1) **Suboptimality of speedup vs condition number lower bound.** The main criticism of the method is the apparent sub-optimality of the derived lower bounds. We provide an intuitive explanation to this point here, refer to [2] for further details. Roughly speaking, the difficulty with a κ -conditioned algorithm for large κ is, that a fixed computer word size cannot accommodate the numbers needed for accurately carrying on the computation. In that sense, the results in [2] tell us that speedup of FFT results in accuracy issues. But these issues could be dealt with by computing with words of size $\Theta(\log \kappa)$, incurring a *bitwise* time complexity cost of $\Omega(\log \kappa)$ per operation. By bitwise time complexity we mean that we count logical binary operations, as measured in fast integer multiplication algorithm design. (See [20] for a groundbreaking recent result in that field, almost matching the trivial $\Omega(\log \kappa)$ bound.) If the lower bound of [3] were tight for $\kappa = \Theta(\log n)$, we could speed up FFT by a factor of κ to obtain a linear time algorithm (in the sense of counting linear algebraic steps),

compensating for the loss of accuracy by emulating words of size $\Theta(\log \log n)$ bits, resulting in bitwise running time of $\tilde{O}(n \log \log n)$.¹² Consequently, if one believes (as we do) that the true bit complexity lower bound of FFT computation should be $\Omega(n \log n)$, then the lower bound in [3] is probably not tight. A major open conjecture in this line of work, which we will come back to in Section 5 is the following tighter speedup vs condition number bound:

Conjecture 1.1 (Informal). *Speedup of FFT by factor of $\kappa > 1$ (counting linear algebraic computation) can only be obtained using $e^{\Omega(\kappa)}$ -conditioned computation.*

(P2) **Fourier perturbation complexity.** Instead of computing the Fourier transform Fx of input $x \in \mathbb{R}^n$, assume we compute a Fourier ε -perturbation thereof, which we define as

$$(\text{Id} + \varepsilon F)x. \quad (1.3)$$

The corresponding matrix $(\text{Id} + \varepsilon F)$ is not orthogonal, but if ε is small then it is almost orthogonal, in the sense that it has condition number $1 + O(\varepsilon)$. In fact, it can be shown (see Appendix B) that a uniformly $(1 + O(\varepsilon))$ -well conditioned algorithm (as defined in detail in Section 2) *can* be used to compute $(\text{Id} + \varepsilon F)$ and, for that matter, any $(1 + O(\varepsilon))$ -conditioned matrix. Now, it is easy to check that the quantity $\Phi(\text{Id} + \varepsilon F)$ is $-\Omega(\varepsilon^2 n \log n)$. Hence, using the arguments in [3] without any change, we get (in the WH case) a lower bound of

$$\Omega(\varepsilon^2 n \log n) \quad (1.4)$$

linear algebraic steps for computing a Fourier ε -perturbation in an algorithm of condition number $(1 + O(\varepsilon))$. The problem is, that it is unlikely that (1.4) is tight. Indeed, assume that a matching algorithm existed. Then we could compute Fx by first computing $(\text{Id} + \varepsilon F)x$ in time $O(\varepsilon^2 n \log n)$ linear algebraic operations, subtracting x from the output (after having stored a copy somewhere) and dividing the result by ε . Now we can drive ε down to $\Theta(1/\sqrt{\log n})$ so that the running time would match the trivial lower bound of linear time. To avoid numerical accuracy problems, we could enlarge the computer word by $\Theta(\log 1/\varepsilon) = \Theta(\log \log n)$ bits, incurring a $\tilde{\Theta}(\log \log n)$ complexity for each addition or multiplication, taking into account logical bit operations.³ The bottom line is an algorithm of running time $\tilde{O}(n \log \log n)$ (counting logical bit operations) for computing the Fourier transform. We don't believe this is possible.

Notice that both problem (P1) and problem (P2) have a similar flavor. They both illustrate that the quasi-entropy method without any adjustment is not strong enough to explain our belief that speeding up Fourier computation by some factor (counting linear algebraic operations) must be offset by the overhead necessary to maintain numerical accuracy. Hence, we hope that solving one problem would shed light on the other.

¹Here, \tilde{O} hides $o(\log \log n)$ factors, arising from the state-of-the-art integer-integer multiplication algorithm [20].

²We say *emulating* because, in a standard computer architecture the word size is constant, but we can emulate “big numbers” in software.

³Notation $\tilde{\Theta}$ here hides $\log \log \log n$ factors, coming from the current state-of-the-art machinery for fast integer multiplication [?]

1.3 Our results

We address problem (P2) in the WH case. The main result in this work (Theorem 4.1) is an exponentially (in ε^{-1}) improvement (over (1.4)) of the form

$$\Omega\left(\frac{n \log n}{\log \varepsilon^{-1}}\right). \quad (1.5)$$

Note that the term $\log \varepsilon^{-1}$ in (1.5) exactly offsets the numerical overhead arising from working with large numbers, as explained in (P2) above. This leads us to believe that the improvement (1.5) is optimal. The main technique used for deriving the improved bound is a new extension of the quasi-entropy method. Our hope is that these techniques (in conjunction with others) could be used to make progress with Conjecture 1.1 in (P1).

2 Notation and Computational Model

The results in this work apply to the WH transform only, which will henceforth be denoted by F . By FFT we mean, the fast WH transform. For the DFT case we believe that a similar method (or extension) should work - see Section 5. For the WH case, all matrix elements are *exactly* in $\{\pm n^{-1/2}\}$. Additionally, the WH transform is symmetric. Hence $F = F^T$, $F^T F = F^2 = \text{Id}$. These algebraic facts make it easier to work with.

We recall the computational model in [3, 2]. We assume a linear algebraic computational model, which means that the state of the machine at each step is a linear transformation of the initial state, which is the input. Therefore, an algorithm running in m steps is expressed as $\mathcal{A} = \{M^{(0)} = \text{Id}, M^{(1)} \dots M^{(m)}\}$, where $M^{(t)}$ is the transformation taking the input x to the t 'th state of the machine. Two consecutive matrices $M^{(t)}$ and $M^{(t-1)}$ differ in either one row indexed i_t (a constant gate) or two rows indexed i_t, i'_t (a rotation). In case of a constant gate, $M^{(t)}$ is obtained by multiplying the i_t 'th row of $M^{(t-1)}$ by a constant $c_t \neq 0$. In case of a rotation, $M^{(t)}$ is given as $R_{i_t, i'_t, \Theta_t} M^{(t-1)}$ for some angle constant Θ_t , where $R_{\cdot, \cdot, \cdot}$ was defined in Section 1. We say “ \mathcal{A} computes X ” if $M^{(m)} = X$.

As argued in [2, 3], the computational model is equivalent via simple reductions, to the so-called linear straight line computation (see [6] and references therein for a definition), except that we assume a *no extra memory* scenario here. This means that the machine state at each moment is a vector in \mathbb{R}^n , represented using exactly n computer words. In particular, we are not allowed to pad the input with zeros, somehow taking advantage of the extra space throughout the computation. Note that FFT work in this model. We refer the reader to [3] for an initial treatment of the *additional memory* regime. This paper shows that there is still much to uncover in the no extra memory regime, and we leave the extra memory regime to a separate research branch.

Throughout M^T is transpose of matrix M , and for invertible M , M^{-T} abbreviates $(M^{-1})^T$. We also borrow from Matlab notation to create submatrices, e.g. $M([1 \ 3 \ 10], [5 \ 6])$ is the submatrix obtained from rows 1, 3, 10 and columns 5, 6. Colon is used to specify the full range of indices, so e.g. $M([1 \ 3], :)$ is the submatrix obtained by stacking rows 1 and 3 on top of each other. For matrices A, B with same number of rows, $[A \ B]$ is the matrix obtained by stacking B to the right of A . All logarithms are assume in base 2.

The condition number $\kappa(M)$ of a nonsingular matrix M is the ratio between its top and bottom singular values, equivalently, the ratio between its spectral norm and that of its inverse. Condi-

tion number is a standard measure used for quantifying numerical robustness of linear algebraic computations, see e.g. chapter 11 of [19].

Definition 2.1. An algorithm \mathcal{A} is (uniformly) κ -well conditioned (for $\kappa \geq 1$) if for all $t \in [m]$ the condition number $\kappa(M^{(t)})$ of the t 'th matrix is bounded by κ .

Recall the definitions of the quasi-entropy function $\Phi(M)$ and the preconditioned version $\Phi_{A,B}(M)$ from Section 1.

Theorem 2.2. [2, 3] If $M^{(t)}$ is obtained from $M^{(t-1)}$ by a constant gate, $\Phi(M^{(t)}) = \Phi(M^{(t-1)})$. If $M^{(t)}$ is obtained from $M^{(t-1)}$ by a rotation acting on rows i_t, i'_t , then

$$|\Phi_{P,Q}(M^{(t)}) - \Phi_{P,Q}(M^{(t-1)})| \leq \|(M^{(t-1)}P)([i_t \ i'_t], :)\|_F \cdot \|((M^{(t-1)})^{-T}Q)([i_t \ i'_t], :)\|_F, \quad (2.1)$$

where $\|\cdot\|_F$ is Frobenius norm.

Note that in particular, if $M^{(t-1)}$ (and hence, also $M^{(t)}$) is κ -well conditioned and P, Q have both spectral norm $O(1)$, then

$$|\Phi(M^{(t)}) - \Phi(M^{(t-1)})| = O(\kappa).$$

For $\varepsilon < 1/2$, a Fourier ε -perturbation is defined as the matrix $(\text{Id} + \varepsilon F)$. The inverse of the ε -Fourier perturbation is given by $\text{Id} - \varepsilon F + Z$, where Z has spectral norm at most $O(\varepsilon^2)$. Using a simple entropy approximation lemma (deferred to Appendix A), it is not difficult to see that the quasi entropy of a Fourier ε -perturbation is

$$\Phi(\text{Id} + \varepsilon F) = -\Omega(\varepsilon^2 n \log n). \quad (2.2)$$

Note that the potential is negative, but this will not matter for the purpose of establishing computational lower bounds. We will assume throughout that $1/\varepsilon = n^{o(1)}$. (Otherwise, computing with numerical accuracy $O(\varepsilon)$ probably requires $\Omega(\log n)$ bits per word, which is not very interesting because then any multiplication or addition requires $\Omega(\log n)$ logical bit operations).

The condition number of a Fourier ε -perturbation is clearly $1 + O(\varepsilon)$. This means that a necessary condition for an algorithm computing a Fourier ε -perturbation is that it is not κ well conditioned for $\kappa = 1 + o(\varepsilon)$. It turns out that there exists an algorithm that is $(1 + O(\varepsilon))$ -well conditioned that computes $(\text{Id} + \varepsilon F)x$ given input x . Refer to Appendix B for details of this simple fact. By Theorem 2.2 we conclude:

Proposition 2.3. If algorithm \mathcal{A} computes $(\text{Id} + \varepsilon F)$ and \mathcal{A} is $(1 + O(\varepsilon))$ -well conditioned, then \mathcal{A} makes $\Omega(\varepsilon^2 n \log n)$ steps.

3 From $\Omega(\varepsilon^2 n \log n)$ to $\Omega(\varepsilon n \log n)$

Assume an algorithm $\mathcal{A} = (\text{Id} = M^{(0)}, M^{(1)}, \dots, M^{(m)} = (\text{Id} + \varepsilon F))$ computes the Fourier ε -perturbation, and that \mathcal{A} is $(1 + O(\varepsilon))$ -well conditioned. The lower bound of $\Omega(\varepsilon^2 n \log n)$ from Proposition 2.3 was obtained by tracking the following sequence of matrix quasi-entropies:

$$0 = \Phi(M^{(0)}), \Phi(M^{(1)}), \dots, \Phi(M^{(m)}) = \Phi(\text{Id} + \varepsilon F) = -\Omega(\varepsilon^2 n \log n).$$

A first step toward improving this bound is by identifying a pair of preconditioning matrices A, B of at most constant spectral norm, for which

$$\Phi_{A,B}(M^{(0)}) = \Phi_{A,B}(\text{Id}) = \pm o(\varepsilon n \log n) \quad \Phi_{A,B}(M^{(m)}) = \Phi_{A,B}(\text{Id} + \varepsilon F) = \pm \Omega(\varepsilon n \log n) .$$

Using Theorem 2.2 (together with the bound on the spectral norm of A, B) implies that for all $t \in [m]$:

$$\left| \Phi_{A,B}(M^{(t)}) - \Phi_{A,B}(M^{(t+1)}) \right| = O(1) .$$

Combining, the implication is that $m = \Omega(\varepsilon n \log n)$. It turns out that this can be achieved by taking $A = \text{Id}$ and $B = F$. It is easy to check that

$$\Phi_{\text{Id},F}(M^{(0)}) = \Phi_{\text{Id},F}(\text{Id}) = O(\sqrt{n} \log n) .$$

We will now prove that $\Phi_{\text{Id},F}(\text{Id} + \varepsilon F) = \Omega(\varepsilon n \log n)$ using the definition of $\Phi_{\text{Id},F}()$ and a straightforward calculation. As before, recall that $(\text{Id} + \varepsilon F)^{-1} = \text{Id} - \varepsilon F + Z$ where Z has spectral norm $O(\varepsilon^2)$.

$$\begin{aligned} \Phi_{\text{Id},F}(\text{Id} + \varepsilon F) &= - \sum_{i=1}^n \sum_{j=1}^n (\text{Id} + \varepsilon F)(i, j) \cdot (F - \varepsilon F^2 + ZF)(j, i) \\ &\quad \log |(\text{Id} + \varepsilon F)(i, j) \cdot (F - \varepsilon F^2 + ZF)(j, i)| . \end{aligned} \tag{3.1}$$

The diagonal summands in the last RHS are numbers in the set $\{x \log |x| : |x| = O(\varepsilon + 1/\sqrt{n})\}$. Hence their total contribution to the sum is (in absolute value)

$$O(n(\varepsilon + 1/\sqrt{n}) \log n) . \tag{3.2}$$

As for the non-diagonal elements, an intuitive analysis uses the fact that the dominant term in $(\text{Id} + \varepsilon F)(i, j) \cdot (F - \varepsilon F^2 + ZF)(j, i)$ is $\varepsilon F(i, j)F(j, i)$. All other terms have order $O(\varepsilon^2)$. For an exact analysis, refer to Appendix A with $\ell = n - 1$ to bound the inner sum of the RHS of (3.1) (excluding the diagonal $j = i$) by $\Omega(\varepsilon \log n)$. Combining,

$$\Phi_{\text{Id},F}(\text{Id} + \varepsilon F) = \Omega(\varepsilon n \log n) , \tag{3.3}$$

as required. The method described in this section can probably not be used to improve on the lower bound of $\Omega(\varepsilon n \log n)$. We need new ideas.

4 From $\Omega(\varepsilon n \log n)$ to $\Omega((n \log n)/\log \varepsilon^{-1})$

We state and prove the main result in this paper.

Theorem 4.1. *If algorithm \mathcal{A} computes $(\text{Id} + \varepsilon F)$ and \mathcal{A} is $(1 + O(\varepsilon))$ -well conditioned, then \mathcal{A} makes $\Omega((n \log n)/\log \varepsilon^{-1})$ steps.*

Proof. In order to achieve the $\Omega((n \log n)/\log \varepsilon^{-1})$ bound, we will use a different version of the preconditioned potential function. For a nonsingular matrix M and two n -by- $2n$ matrices P, Q , we define

$$\begin{aligned} \hat{\Phi}_{P,Q}(M) &= - \sum_{i=1}^n \sum_{j=1}^n ((MP)(i, j)(M^{-T}Q)(i, j) + (MP)(i, j+n)(M^{-T}Q)(i, j+n)) \\ &\quad + \log |(MP)(i, j)(M^{-T}Q)(i, j) + (MP)(i, j+n)(M^{-T}Q)(i, j+n)| \end{aligned}$$

We will use $\hat{\Phi}$ in conjunction with the preconditioners:

$$P = [\text{Id}, -F]$$

$$Q = [F, \text{Id}] .$$

We aim to prove that a rotation can change the potential by at most $O(\varepsilon \log \varepsilon^{-1})$. Let t be such that $M^{(t+1)}$ is obtained from $M^{(t)}$ by a rotation. (It is clear to see that a constant gate does not change the potential.) Therefore we can assume that $M^{(t+1)} = R_{i,i',\Theta} M^{(t)}$ for some row indices i, i' and angle Θ . Without loss of generality we can assume that $i = 1, i' = 2$, in other words that the rotation affects rows $i = 1, 2$ only. First, one can notice that since $\kappa(M^{(t)}) = 1 + O(\varepsilon)$, we can write

$$M^{(t)} = U + \Delta \tag{4.1}$$

$$(M^{(t)})^{-T} = U + \Gamma \tag{4.2}$$

where U is an orthogonal matrix, and Δ, Γ have spectral norm $O(\varepsilon)$.

For the purpose of tracking the change in potential $\hat{\Phi}_{P,Q}$ we can ignore the contribution to the potential coming from rows $i > 2$. Let $V = UF, \Xi = \Delta F, \Lambda = \Gamma F$, and $L(x) = x \log |x|$ for any real x . If $\hat{\Phi}_{|1,2}$ denotes the contribution to $\hat{\Phi}_{P,Q}(M^{(t)})$ coming from rows $i = 1, 2$, then we can now express it as:

$$\begin{aligned} & - \sum_{j=1}^n L((U(1,j) + \Delta(1,j))(V(1,j) + \Lambda(1,j)) - (V(1,j) + \Xi(1,j))(U(1,j) + \Gamma(1,j))) \\ & - \sum_{j=1}^n L((U(2,j) + \Delta(2,j))(V(2,j) + \Lambda(2,j)) - (V(2,j) + \Xi(2,j))(U(2,j) + \Gamma(2,j))) . \end{aligned}$$

Notice now that the term $U(1,k)V(1,k)$ disappears from the first row, and $U(2,k)V(2,k)$ from the second. Hence,

$$\begin{aligned} \hat{\Phi}_{|1,2} = & - \sum_{j=1}^n L(U(1,j)\Lambda(1,j) + V(1,j)\Delta(1,j) + \Delta(1,j)\Lambda(1,j) \\ & - V(1,j)\Gamma(1,j) - U(1,j)\Xi(1,j) - \Gamma(1,j)\Xi(1,j)) \\ & - \sum_{j=1}^n L(U(2,j)\Lambda(2,j) + V(2,j)\Delta(2,j) + \Delta(2,j)\Lambda(2,j) \\ & - V(2,j)\Gamma(2,j) - U(2,j)\Xi(2,j) - \Gamma(2,j)\Xi(2,j)) . \end{aligned} \tag{4.3}$$

Now let

$$r(j) = \sqrt{U(1,j)^2 + V(1,j)^2 + U(2,j)^2 + V(2,j)^2} \tag{4.4}$$

$$\rho(j) = \sqrt{\Delta(1,j)^2 + \Xi(1,j)^2 + \Delta(2,j)^2 + \Xi(2,j)^2 + \Gamma(1,j)^2 + \Lambda(1,j)^2 + \Gamma(2,j)^2 + \Lambda(2,j)^2} . \tag{4.5}$$

Note that by our construction we have:

$$\sum_{j=1}^n r(j)^2 = 4 \quad \sum_{j=1}^n \rho(j)^2 = O(\varepsilon) \quad (4.6)$$

where the left identity is by orthogonality of U, V and the right bound is by the spectral bound of $O(\varepsilon)$ on $\Delta, \Gamma, \Delta, \Xi$ and Λ . Dividing and multiplying by $r^2(j)$, (4.3) now becomes

$$\begin{aligned} \hat{\Phi}_{|1,2} &= \sum_{j=1}^n L \left(r^2(j) \left(\frac{U(1,j)\Lambda(1,j)}{r^2(j)} + \frac{V(1,j)\Delta(1,j)}{r^2(j)} + \frac{\Delta(1,j)\Lambda(1,j)}{r^2(j)} \right. \right. \\ &\quad \left. \left. - \frac{V(1,j)\Gamma(1,j)}{r^2(j)} - \frac{U(1,j)\Xi(1,j)}{r^2(j)} - \frac{\Gamma(1,j)\Xi(1,j)}{r^2(j)} \right) \right) \\ &+ \sum_{j=1}^n L \left(r^2(j) \left(\frac{U(2,j)\Lambda(2,j)}{r^2(j)} + \frac{V(2,j)\Delta(2,j)}{r^2(j)} + \frac{\Delta(2,j)\Lambda(2,j)}{r^2(j)} \right. \right. \\ &\quad \left. \left. - \frac{V(2,j)\Gamma(2,j)}{r^2(j)} - \frac{U(2,j)\Xi(2,j)}{r^2(j)} - \frac{\Gamma(2,j)\Xi(2,j)}{r^2(j)} \right) \right) \end{aligned} \quad (4.7)$$

For simplicity of notation, for rows $i = 1, 2$ and any column j we define:

$$\begin{aligned} X(i, j) &= \frac{U(i, j)\Lambda(i, j)}{r^2(j)} + \frac{V(i, j)\Delta(i, j)}{r^2(j)} + \frac{\Delta(i, j)\Lambda(i, j)}{r^2(j)} \\ &- \frac{V(i, j)\Gamma(i, j)}{r^2(j)} - \frac{U(i, j)\Xi(i, j)}{r^2(j)} - \frac{\Gamma(i, j)\Xi(i, j)}{r^2(j)}. \end{aligned} \quad (4.8)$$

Now recall that $M^{(t+1)}$ is obtained from $M^{(t)}$ by a rotation matrix $R_{1,2,\Theta}$, affecting the first two rows only. Therefore, from (4.1) and (4.2) we have

$$\begin{aligned} M^{(t+1)} &= U' + \Delta' \\ (M^{(t+1)})^{-T} &= U' + \Gamma' \end{aligned}$$

where $U' = R_{1,2,\Theta}U$, $\Delta' = R_{1,2,\Theta}\Delta$, $\Gamma' = R_{1,2,\Theta}\Gamma$. Similarly, we define $V' = R_{1,2,\Theta}V$, $\Xi' = R_{1,2,\Theta}\Xi$, $\Lambda' = R_{1,2,\Theta}\Lambda$ as the ‘post-rotation’ version of the corresponding variables. We can also define $r'(j), \rho'(j)$ similarly to (4.4) and (4.5), but with the post-rotation variables. However clearly $r(j) = r'(j)$ and $\rho(j) = \rho'(j)$ because rotation is an isometry, so $U(1, j)^2 + U(2, j)^2 = U'(1, j)^2 + U'(2, j)^2$ and similarly for the other components in the RHS's of (4.4) and (4.5). The ultimate goal is to compare the corresponding potentials $\hat{\Phi}_{|1,2}$ and $\hat{\Phi}'_{|1,2}$, where $\hat{\Phi}'_{|1,2}$ is obtained as in (4.7), but using the post-rotation variables.

Now consider the expression $X(1, j) + X(2, j)$ for fixed j . This expression can be viewed as sum of (scaled) inner products. For example, the first inner product is

$$\frac{\langle (U(1, j), U(2, j)), (\Lambda(1, j), \Lambda(2, j)) \rangle}{r^2(j)}.$$

Hence,

$$X(1, j) + X(2, j) = X'(1, j) + X'(2, k), \quad (4.9)$$

where $X'(i, j)$ is obtained as in (4.8), but using the post-rotation variables. Indeed planar inner products are not affected by a planar rotation. We are now finally in a position to compare $\hat{\Phi}_{|1,2}$ with $\hat{\Phi}'_{|1,2}$.

$$|\hat{\Phi}_{|1,2} - \hat{\Phi}'_{|1,2}| = \left| \sum_{j=1}^n (r(j)^2 X(1, j) \log(|r(j)^2 X(1, j)|) + r(j)^2 X(2, j) \log(|r(j)^2 X(2, j)|)) \right. \\ \left. - \sum_{j=1}^n (r(j)^2 X'(1, j) \log(r(j)^2 X'(1, j)) + r(j)^2 X'(2, j) \log(r(j)^2 X'(2, j))) \right| \quad (4.10)$$

$$= \left| \sum_{j=1}^n (r(j)^2 \log(|r(j)^2|) (X(1, j) + X(2, j) - X'(1, j) - X'(2, j))) \right| \quad (4.11)$$

$$+ \sum_{j=1}^n (r(j)^2 (L(X(1, j)) + L(X(2, j)) - L(X'(1, j)) - L(X'(2, j)))) \left| \right. \\ = \left| \sum_{j=1}^n (r(j)^2 (L(X(1, j)) + L(X(2, j)) - L(X'(1, j)) - L(X'(2, j)))) \right| \\ \leq \sum_{j=1}^n r(j)^2 |(L(X(1, j)) + L(X(2, j)) - L(X'(1, j)) - L(X'(2, j)))|, \quad (4.12)$$

where the first equality is application of the rule $\log(xy) = \log x + \log y$, the second is from (4.9) and the inequality is application of the triangle inequality. Clearly, $|U(i, j)|, |V(i, j)| \leq r(j)$ and $|\Lambda(i, j)|, |\Delta(i, j)|, |\Gamma(i, j)|, |\Xi(i, j)| \leq \rho(j)$ for $i = 1, 2$ and $j \in [n]$. Therefore by definition of $X(i, j)$, $|X(i, j)| \leq 6\rho(j)/r(j)$. A similar bound holds for $X'(i, j)$. For fixed j , the inner sum of (4.12) is hence bounded above by

$$r(j)^2 \left(2 \max_{|x| \leq 6\frac{\rho(j)}{r(j)}} L(x) - 2 \min_{|y| \leq 6\frac{\rho(j)}{r(j)}} L(y) \right).$$

Let e be the natural logarithm basis. For nonnegative a such that $a \leq e^{-1}$, $\max_{x: |x| \leq a} L(x) = -|a| \log |a|$ and $\min_{x: |x| \leq a} L(x) = |a| \log |a|$. For $a > e^{-1}$, $\max_{x: |x| \leq a} L(x) \leq a(3 + \log a)$ and $\min_{x: |x| \leq a} L(x) \geq -a(3 + \log a)$. We will now define the following subsets of column indices, indexed by an integer h :

$$J_h = \begin{cases} \left\{ j : 6\frac{\rho(j)}{r(j)} \leq \varepsilon \right\} & h = 0 \\ \left\{ j : 2^{h-1}\varepsilon \leq 6\frac{\rho(j)}{r(j)} \leq \min\{2^h\varepsilon, e^{-1}\} \right\} & 1 \leq h \leq \lceil -\log(\varepsilon e) \rceil \\ \left\{ j : \max\{e^{-1}, 2^{h-1}\varepsilon\} \leq 6\frac{\rho(j)}{r(j)} \leq 2^h\varepsilon \right\} & h > \lceil -\log(\varepsilon e) \rceil \end{cases}.$$

Splitting the sum (4.12) and applying our analysis of the function $L(x)$, we get

$$\begin{aligned}
|\hat{\Phi}_{|1,2} - \hat{\Phi}'_{|1,2}| &\leq \sum_{h \geq 0} \sum_{j \in J_h} r(j)^2 \left(2 \max_{|x| \leq 6 \frac{\rho(j)}{r(j)}} L(x) - 2 \min_{|y| \leq 6 \frac{\rho(j)}{r(j)}} L(y) \right) \\
&\leq -4 \sum_{j \in J_0} r(j)^2 \varepsilon \log \varepsilon - 4 \sum_{h=1}^{\lceil -\log(\varepsilon e) \rceil} \sum_{j \in J_h} r(j)^2 2^h \varepsilon \log(2^h \varepsilon) \\
&\quad + 4 \sum_{h > \lceil -\log(\varepsilon e) \rceil} \sum_{j \in J_h} r(j)^2 2^h \varepsilon (3 + \log(2^h \varepsilon)) .
\end{aligned} \tag{4.13}$$

Let us now bound $\sum_{j \in J_h} r(j)^2$ for all h . For $h = 0$ we can simply recall (4.6) to get the trivial

$$\sum_{j \in J_0} r(j)^2 \leq 4 . \tag{4.14}$$

As for $h \geq 1$, from the definition of J_h , we get that for all $j \in J_h$:

$$r^2(j) \leq \frac{36\rho^2(j)}{2^{2h-2}\varepsilon^2} .$$

Therefore,

$$\sum_{j \in J_h} r^2(j) \leq \sum_{j=1}^n \frac{36\rho^2(j)}{2^{2h-2}\varepsilon^2} \leq C 2^{-2h} , \tag{4.15}$$

where the rightmost bound is by (4.6) and C is proportional to the constant hiding in the $O(\cdot)$ -notation there. Plugging the bounds (4.14) and (4.15) in (4.13) gives:

$$\begin{aligned}
|\hat{\Phi}_{|1,2} - \hat{\Phi}'_{|1,2}| &\leq -4\varepsilon \log \varepsilon - 4 \sum_{h=1}^{\lceil -\log(\varepsilon e) \rceil} C 2^{-2h} 2^h \varepsilon \log(2^h \varepsilon) \\
&\quad + 4 \sum_{h > \lceil -\log(\varepsilon e) \rceil} C 2^{-2h} 2^h \varepsilon (3 + \log(2^h \varepsilon)) \\
&= O(-\varepsilon \log \varepsilon) .
\end{aligned} \tag{4.16}$$

In order to complete the proof, we will show that $\hat{\Phi}_{P,Q}(Id) = 0$ and $\Phi_{P,Q}(\hat{Id} + \varepsilon F) = O(\varepsilon n \log n)$.

$$\begin{aligned}
\hat{\Phi}_{P,Q}(Id) &= - \sum_{i=1}^n \sum_{j=1}^n ((P)(i,j)(Q)(i,j) + (P)(i,j+n)(Q)(i,j+n)) \\
&\quad + \log |(P)(i,j)(Q)(i,j) + (P)(i,j+n)(Q)(i,j+n)|
\end{aligned}$$

By the definition of P and Q , it holds that $P_{i,j}Q_{i,j} = -P_{i,j+n}Q_{i,j+n}$ and hence $\hat{\Phi}_{P,Q}(Id) = 0$. We now prove that $\hat{\Phi}_{P,Q}(Id + \varepsilon F) = O(\varepsilon n \log(n))$. We remind the reader that $(Id + \varepsilon F)^{-1} = Id - \varepsilon F + Z$ where Z has spectral norm $O(\varepsilon^2)$.

$$\begin{aligned}
\hat{\Phi}_{P,Q}(Id + \varepsilon F) &= \\
&- \sum_{i=1}^n \sum_{j=1}^n ((Id + \varepsilon F)(i,j)(F - \varepsilon F^2 + Z^T F)(i,j) + (-(F + \varepsilon F^2))(i,j)(Id - \varepsilon F + Z^T)(i,j)) \\
&+ \log |((Id + \varepsilon F)(i,j)(F - \varepsilon F^2 + Z^T F)(i,j) + (-(F + \varepsilon F^2))(i,j)(Id - \varepsilon F + Z^T)(i,j))|
\end{aligned} \tag{4.17}$$

The diagonal elements in the last sum are of the form $\{x \log |x| : |x| = O(\varepsilon + \frac{1}{\sqrt{n}})\}$. Therefore their contribution to the sum is $O(n(\varepsilon + \sqrt{n}^{-1}) \log n)$.

The main thing to note about the off-diagonal elements in the sum (4.17), as complicated as it may seem, is that products of the form $\text{Id}(i, j)F(i, j)$ always appear together with both a $+$ and a $-$ sign, hence cancel each other out. The bottleneck is elements multiplied by ε exactly once. (Note that our bound on the spectral norm of Z involves ε^2 , hence elements such as $Z(i, j)$ or $(ZF)(i, j)$ are not part of the bottleneck.) These bottleneck elements give the theorem's required bound. The remaining elements are 'noise', which can be accounted for using Lemma A.1 in Appendix A for each row i . To apply the lemma, take $\ell = n - 1$ (accounting for all elements in the row except diagonal), x 's coordinates are the elements $2\varepsilon F(i, j)^2$ (for $j \neq i$) and the vector y is given by collecting the remaining elements, namely:

$$-\varepsilon^2 F(i, j)(F^2)(i, j) + \varepsilon F(i, j)(Z^T F)(i, j) - F(i, j)Z(i, j) - \varepsilon^2 (F^2)(i, j)F(i, j) - \varepsilon (F^2)(i, j)Z^T(i, j) .$$

The resulting estimation is:

$$\hat{\Phi}_{P,Q}(\text{Id} + \varepsilon F) = \Omega(\varepsilon n \log(n/\varepsilon)) . \quad (4.18)$$

(Recall that we assumed $1/\varepsilon$ is $n^{o(1)}$, hence $\log(n/\varepsilon) = \Theta(\log n)$.) Combining this with the bound $O(-\varepsilon \log \varepsilon)$ on the change of $\hat{\Phi}$ at each step concludes the proof of the theorem. This concludes the proof of the theorem. \square

5 Future Work

The main technique used here was extending the quasi-entropy function. The extension was 'tailored' for the Fourier (WH) ε -perturbation, but we believe that a similar analysis is doable for DFT as well. In fact, one can significantly extend the method giving rise to much freedom in proving bounds for other interesting problems. A reasonable way to further extend the quasi-entropy function is to define

$$\Phi_{[A_1, \dots, A_k], [B_1, \dots, B_k]}^{(k)}(M) = \sum_{i=1}^n \sum_{j=1}^n \left(\sum_{p=1}^k (MA_p)(i, j)(M^{-T} B_p)(i, j) \right) \log \left| \sum_{p=1}^k (MA_p)(i, j)(M^{-T} B_p)(i, j) \right| ,$$

where M is nonsingular and the A_p 's, B_p 's are square $n \times n$. The function $\hat{\Phi}$ we used corresponds exactly to $k = 2$.

The main hope is that this technique, or some further extension, could be used to make progress on Conjecture 1.1. Another open problem is to match the lower bound of Theorem 4.1, using a $(1 + O(\varepsilon))$ -well conditioned algorithm (without extra memory). We show in the appendix that a quadratic time algorithm exists, leaving quite a large gap.

References

- [1] Nir Ailon. A lower bound for fourier transform computation in a linear model over 2x2 unitary gates using matrix entropy. *Chicago J. Theor. Comput. Sci.*, 2013, 2013.
- [2] Nir Ailon. Tighter Fourier transform lower bounds. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 14–25, 2015.
- [3] Nir Ailon. An $\Omega((n \log n)/r)$ lower bound for Fourier transform computation in the r -well conditioned model. *TOCT*, 8(1):4, 2016.
- [4] Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging LAPACK’s least-squares solver. *SIAM J. Sci. Comput.*, 32(3):1217–1236, April 2010.
- [5] Bryce E Bayer. Image processing method using a collapsed Walsh-Hadamard transform, October 22 1985. US Patent 4,549,212.
- [6] Michael Ben-Or. Lower bounds for algebraic computation trees. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC ’83, pages 80–86, 1983.
- [7] Yaakoub Berrouche and Raïs Elhadi Bekka. Improved multiple description wavelet based image coding using Hadamard Transform. *AEU-International Journal of Electronics and Communications*, 68(10):976–982, 2014.
- [8] Eric Blais. Testing juntas nearly optimally. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 151–158. ACM, 2009.
- [9] Eric Blais. *Testing properties of Boolean functions*. PhD thesis, US Army, 2012.
- [10] Christos Boutsidis and Alex Gittens. Improved matrix algorithms via the subsampled randomized Hadamard transform. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1301–1340, 2013.
- [11] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.
- [12] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
- [13] Ronald De Wolf. A brief introduction to Fourier analysis on the boolean cube. *Theory of Computing, Graduate Surveys*, 1:1–20, 2008.
- [14] JA Decker. Hadamard–transform image scanning. *Applied optics*, 9(6):1392–1395, 1970.
- [15] Douglas F Elliott. *Handbook of digital signal processing: engineering applications*. Academic press, 2013.
- [16] B.J. Fino and V.R. Algazi. Unified matrix treatment of the fast Walsh-Hadamard transform. *IEEE Transactions on Computers*, 25(11):1142–1146, 1976.

- [17] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas [combinatorial property testing]. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 103–112. IEEE, 2002.
- [18] Martin Fürer. Faster integer multiplication. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 57–66, New York, NY, USA, 2007. ACM.
- [19] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 4th edition, 2013.
- [20] David Harvey, Joris Van Der Hoeven, and Grégoire Lecerf. Even faster integer multiplication. *arXiv preprint arXiv:1407.3360*, 2014.
- [21] Daniel J Lum, Samuel H Knarr, and John C Howell. Fast Hadamard transforms for compressive sensing of joint systems: measurement of a 3.2 million-dimensional bi-photon probability distribution. *Optics express*, 23(21):27636–27649, 2015.
- [22] Stéphane Mallat. *A wavelet tour of signal processing*. Academic press, 1999.
- [23] Yishay Mansour. Learning boolean functions via the Fourier transform. In *Theoretical advances in neural computation and learning*, pages 391–424. Springer, 1994.
- [24] Gunnar Martinsson, Adrianna Gillman, Edo Liberty, Nathan Halko, Vladimir Rokhlin, Sijia Hao, Yoel Shkolnisky, Patrick Young, Joel Tropp, Mark Tygert, et al. Randomized methods for computing the singular value decomposition (SVD) of very large matrices. In *Workshop on Algorithms for Modern Massive Data Sets, Palo Alto*, 2010.
- [25] Jacques Morgenstern. Note on a lower bound on the linear complexity of the fast Fourier transform. *J. ACM*, 20(2):305–306, April 1973.
- [26] Christos H. Papadimitriou. Optimality of the fast Fourier transform. *J. ACM*, 26(1):95–102, January 1979.
- [27] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 143–152. IEEE, 2006.
- [28] Martin Vetterli. Multi-dimensional sub-band coding: some theory and algorithms. *Signal processing*, 6(2):97–112, 1984.
- [29] Martin Vetterli, Henri J Nussbaumer, et al. Simple FFT and DCT algorithms with reduced number of operations. *Signal processing*, 6(4):267–278, 1984.

A A Useful Lemma on Entropy with Noise

Lemma A.1. *For any large enough integer ℓ , for any $x \in (\mathbb{R}^+)^{\ell}$ such that $\|x\|_1 \leq 1$, $\|x\|_{\infty} \leq 4\|x\|_1/\ell$ and for any $y \in \mathbb{R}^{\ell}$ such that $\|y\|_1 \leq C\|x\|_1$ (for some global C):*

$$-\sum_{i=1}^{\ell} (x_i + y_i) \log |x_i + y_i| \geq \|x\|_1 \log \frac{\ell}{\|x\|_1} - 10. \quad (\text{A.1})$$

Proof. Let I_{big} denote $\{i \in [\ell] : |y_i| \geq x_i/2\}$, and let I_{small} denote $[\ell] \setminus I_{\text{big}}$. By the monotonicity of the function $-x \log |x|$ in the range $x \in [0, e^{-1}]$, and by choosing ℓ large enough so that $|x_i + y_i| \leq e^{-1}$ for $i \in I_{\text{small}}$,⁴

$$\begin{aligned} - \sum_{i \in I_{\text{small}}} (x_i + y_i) \log |x_i + y_i| &\geq - \sum_{i \in I_{\text{small}}} \frac{x_i}{2} \log \frac{x_i}{2} \\ &\geq - \sum_{i \in I_{\text{small}}} \frac{x_i}{2} \log(4\|x\|_1/\ell) \end{aligned}$$

We now estimate $\sum_{i \in I_{\text{small}}} x_i = 1 - \sum_{i \in I_{\text{big}}} x_i$. By definition of I_{big} , $\sum_{i \in I_{\text{big}}} x_i \leq \sum_{i \in I_{\text{big}}} 2y_i \leq 2C\|x\|_1$ which is at most $\|x\|_1/2$ (assuming C small enough). Therefore, $\sum_{i \in I_{\text{small}}} x_i \geq \|x\|_1/2$ and consequently

$$- \sum_{i \in I_{\text{small}}} (x_i + y_i) \log |x_i + y_i| \geq \frac{1}{4} \|x\|_1 \log(\ell/(4\|x\|_1)) . \quad (\text{A.2})$$

We now bound the contribution from I_{big} . For each $i \in I_{\text{big}}$, by our construction, $|x_i + y_i| \leq 3|y_i|$. Therefore,

$$\begin{aligned} \left| - \sum_{i \in I_{\text{big}}} (x_i + y_i) \log |x_i + y_i| \right| &\leq - \sum_{i \in I_{\text{big}}} |x_i + y_i| \log |x_i + y_i| \\ &\leq - \sum_{i \in I_{\text{big}}} 3|y_i| \log |3y_i| \\ &\leq 3\|y\|_1 \log \ell - 3\|y\|_1 \log(3\|y\|_1) \\ &\leq 3C\|x\|_1 \log \ell - 3C\|x\|_1 \log(3C\|x\|_1) \\ &= 3C\|x\|_1 \log(\ell/(3C\|x\|_1)) , \end{aligned} \quad (\text{A.3})$$

where the third inequality is by well known properties of the Shannon entropy function for non-negative vectors, and the fourth inequality assumes that C is small enough so that $3C\|x\|_1 \leq e^{-1}$ (recall that $-x \log x$ is monotone increasing in $[0, e^{-1}]$). Choosing C small enough, and combining (A.2) with (A.3) gives the desired result. \square

B Fourier ε -Perturbation Can be Computed by a $(1 + O(\varepsilon))$ -Well Conditioned Algorithm

For completeness, we prove the simple fact stated in the section title. By the SVD theorem, the matrix $(\text{Id} + \varepsilon F)$ can be written as a product of three matrices $U\Sigma V^T$, where U and V are real orthogonal and Σ is diagonal nonnegative, with the elements on the diagonal in the range $[1 - \varepsilon, 1 + \varepsilon]$. Therefore, to compute $(\text{Id} + \varepsilon F)x$ we can first compute $V^T x$, using the well known fact that any real orthogonal matrix is a composition of $O(n^2)$ rotations (see Chapter 5 on *Givens* rotations in [19]). Continuing from there, we can compute $\Sigma V^T x$ using constant gates, one per coordinate. Finally, we get $U\Sigma V^T x$ by decomposing U as $O(n^2)$ rotations. Clearly this computation is $(1 + O(\varepsilon))$ -well conditioned.

⁴By the construction, such ℓ can be chosen independently of x, y